ALGORITHM xxx: MINRES-QLP for Singular Symmetric and Hermitian Linear Equations and Least-Squares Problems

SOU-CHENG T. CHOI
University of Chicago/Argonne National Laboratory
and
MICHAEL A. SAUNDERS
Stanford University

We describe algorithm MINRES-QLP and its FORTRAN 90 implementation for solving symmetric or Hermitian linear systems or least-squares problems. If the system is singular, MINRES-QLP computes the unique minimum-length solution (also known as the pseudoinverse solution), which generally eludes MINRES. In all cases, it overcomes a potential instability in the original MINRES algorithm. A positive-definite preconditioner may be supplied. Our FORTRAN 90 implementation illustrates a design pattern that allows users to make problem data known to the solver but hidden and secure from other program units. In particular, we circumvent the need for reverse communication. While we focus here on a FORTRAN 90 implementation, we also provide and maintain MATLAB versions of MINRES and MINRES-QLP.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods)*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing; statistical software*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN program units*

General Terms: Algorithms

Additional Key Words and Phrases: Krylov subspace method, Lanczos process, conjugate-gradient method, singular least-squares, linear equations, minimum-residual method, pseudoinverse solution, ill-posed problem, regression, sparse matrix, data encapsulation

Version 18 of January 15, 2013

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357; National Science Foundation grant CCR-0306662; Office of Naval Research grants N00014-02-1-0076 and N00014-08-1-0191; and U.S. Army Research Laboratory through the Army High Performance Computing Research Center.

Authors' addresses: S.-C. T. Choi, Computation Institute, University of Chicago, Chicago, IL 60637; email: sctchoi@uchicago.edu; M. A. Saunders, Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4121; email: saunders@stanford.edu.

1. INTRODUCTION

MINRES-QLP [Choi 2006; Choi et al. 2011] is a Krylov subspace method for computing the minimum-length and minimum-residual solution (also known as the pseudoinverse solution) x to the following linear systems or least-squares (LS) problems:

$$\text{solve } Ax = b, \tag{1}$$

$$\text{minimize } \|x\|_2 \quad \text{s.t.} \quad Ax = b, \tag{2}$$

$$\text{minimize } \|x\|_2 \quad \text{s.t.} \quad x \in \arg\min_x \|Ax - b\|_2, \tag{3}$$

where A is an $n \times n$ symmetric or Hermitian matrix and b is a real or complex n-vector. Problems (1) and (2) are treated as special cases of (3). The matrix A is usually large and sparse, and it may be singular.[1] It is defined by means of a user-written subroutine `Aprod`, whose function is to compute the product $y = Av$ for any given vector v.

Let x_k be the solution estimate associated with MINRES-QLP's kth iteration, with residual vector $r_k = b - Ax_k$. Without loss of generality, we define $x_0 = 0$. MINRES-QLP provides recurrent estimates of $\|x_k\|$, $\|r_k\|$, $\|Ar_k\|$, $\|A\|$, $\text{cond}(A)$, and $\|Ax_k\|$, which are used in the stopping conditions.

Other iterative methods specialized for symmetric systems $Ax = b$ are the conjugate-gradient method (CG) [Hestenes and Stiefel 1952], SYMMLQ and MINRES [Paige and Saunders 1975], and SQMR [Freund and Nachtigal 1994]. Each method requires one product Av_k at each iteration for some vector v_k. CG is intended for positive-definite A, whereas the other solvers allow A to be indefinite.

If A is singular, SYMMLQ requires the system to be consistent, whereas MINRES returns an LS solution for (3) but generally not the min-length solution; see [Choi 2006; Choi et al. 2011] for examples. SQMR without preconditioning is mathematically equivalent to MINRES but could fail on a singular problem. To date, MINRES-QLP is probably the most suitable CG-type method for solving (3).

In some cases the more established symmetric methods may still be preferable.

(1) If A is positive definite, CG minimizes the energy norm of the error $\|x - x_k\|_A$ in each Krylov subspace and requires slightly less work per iteration. However, CG, MINRES, and MINRES-QLP do reduce $\|x - x_k\|_A$ and $\|x - x_k\|$ monotonically. Also, MINRES and MINRES-QLP often reduce $\|r_k\|$ to the desired level significantly sooner than does CG, and the backward error for each x_k decreases monotonically. (See Section 2.4 and [Fong 2011; Fong and Saunders 2012].)

(2) If A is indefinite but $Ax = b$ is consistent (e.g., if A is nonsingular), SYMMLQ requires slightly less work per iteration, and it reduces the error norm $\|x - x_k\|$ monotonically. MINRES and MINRES-QLP *usually* reduce $\|x - x_k\|$ [Fong 2011; Fong and Saunders 2012].

(3) If A is indefinite and well-conditioned and $Ax = b$ is consistent, MINRES might be preferable to MINRES-QLP because it requires the same number of iterations but slightly less work per iteration.

[1] A further input parameter σ (a real shift parameter) causes MINRES-QLP to treat "A" as if it were $A - \sigma I$. For example, "singular A" really means that $A - \sigma I$ is singular.

(4) MINRES and MINRES-QLP require a preconditioner to be positive definite. SQMR might be preferred if A is indefinite and an effective indefinite preconditioner is available.

MINRES-QLP has two phases. Iterations start in the *MINRES phase* and transfer to the *MINRES-QLP phase* when a subproblem (see (8) below) becomes ill-conditioned by a certain measure. If every subproblem is of full rank and well-conditioned, the problem can be solved entirely in the MINRES phase, where the cost per iteration is essentially the same as for MINRES. In the MINRES-QLP phase, one more work vector and $5n$ more multiplications are used per iteration.

MINRES-QLP described here is implemented in FORTRAN 90 for real double-precision problems. It contains no machine-dependent constants and does not need to use features such as polymorphism from FORTRAN 2003 or 2008. It requires an auxiliary subroutine `Aprod` and, if a preconditioner is supplied, a second subroutine `Msolve`. Since FORTRAN 90 contains the intrinsic `COMPLEX` data type, our implementation is also adapted for complex problems. Precision other than double can be handily obtained by supplying different values to the data attribute `KIND`. The program can be compiled with FORTRAN 90 and FORTRAN 95 compilers such as `f90`, `f95`, `g95`, and `gfortran`. We also have a MATLAB implementation, which is capable of solving both real and complex problems readily. All versions are available for download at [SOL].

Table I lists the main notation used.

Table I. Key notation.

$\|\cdot\|$	matrix or vector two-norm
$\bar{A}$	$\bar{A} = A - \sigma I$ (see also σ below)
cond(A)	condition number of A with respect to two-norm $= \frac{\max \|\lambda_i\|}{\min_{\lambda_i \neq 0} \|\lambda_i\|}$
e_i	ith unit vector
ℓ	index of the last Lanczos iteration when $\beta_{\ell+1} = 0$
n	order of A
null(A)	null space of A defined as $\{x \in \mathbb{R}^n \mid Ax = 0\}$
range(A)	column space of A defined as $\{Ax \mid x \in \mathbb{R}^n\}$
^{T}	(right superscript to a vector or a matrix) transpose
$x^\dagger$	unique minimum-length least-squares solution of problem (3)
$\mathcal{K}_k(A, b)$	kth Krylov subspace defined as span$\{b, Ab, \ldots, A^{k-1}b\}$
ε	machine precision
σ	scalar shift to diagonal of A

1.1 Least-Squares Methods

Further existing methods that could be applied to (3) are CGLS and LSQR [Paige and Saunders 1982a; 1982b], LSMR [Fong and Saunders 2011], and GMRES [Saad and Schultz 1986], all of which reduce $\|r_k\|$ monotonically. The first three methods would require two products Av_k and Au_k each iteration and would be generating points in less favorable subspaces. GMRES requires only products Av_k and could use any nonsingular (possibly indefinite) preconditioner. It needs increasing storage and work each iteration, perhaps requiring restarts, but it could be more effective

Table II. Comparison of various least-squares solvers on $n \times n$ systems (3). Storage refers to memory required by working vectors in the solvers. Work counts number of floating-point multiplications. On inconsistent systems, all solvers below except MINRES and GMRES with restart parameter m return the minimum-length LS solution (assuming no preconditioner).

Solver	Storage	Work per Iteration	Products per Iteration	Systems to Solve per Iteration with Preconditioner
MINRES	$7n$	$9n$	1	1
MINRES-QLP	$7n$–$8n$	$9n$–$14n$	1	1
GMRES(m)	$(m+2)n$	$(m+3+1/m)n$	1	1
CGLS	$4n$	$5n$	2	2
LSQR	$5n$	$8n$	2	2
LSMR	$6n$	$9n$	2	2

than MINRES or MINRES-QLP (and the other solvers) if few total iterations were required. Table II summarizes the computational requirements of each method.

1.2 Regularization

We do not discourage using CGLS, LSQR, or LSMR if the goal is to regularize an ill-posed problem using a small damping factor $\lambda > 0$ as follows:

$$\min_x \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|. \tag{4}$$

However, this approach destroys the original problem's symmetry. The normal equation of (4) is $(A^2 + \lambda^2 I)x = Ab$, which suggests that a diagonal shift to A may well serve the same purpose in some cases. For symmetric positive-definite A, $\bar{A} = A - \sigma I$ with $\sigma < 0$ enjoys a smaller condition number. When A is indefinite, a good choice of σ may not exist, for example, if the eigenvalues of A were symmetrically positioned around zero. When this symmetric form is applicable, it is convenient in MINRES and MINRES-QLP; see (3), (5), and (15). We also remark that MINRES and MINRES-QLP produce good estimates of the largest and smallest singular values of $\bar{A}$ (via diagonal values of R_k or L_k in (7) and (11); see [Choi et al. 2011, Section 4]).

Three other regularization tools in the literature (see [Golub and Van Loan 1996, Sections 12.1.1-12.1.3] and [Hansen 1998]) are LSQI, cross-validation, and L-curve. LSQI involves solving a nonlinear equation and is not immediately compatible with the Lanczos framework. Cross-validation takes one row out at a time and thus does not preserve symmetry. The L-curve approach for a CG-type method takes iteration k as the regularization parameter [Hansen 1998, Chapter 8] if both $\|r_k\|$ and $\|x_k\|$ are monotonic. By design, $\|r_k\|$ is monotonic in MINRES and MINRES-QLP, and so is $\|x_k\|$ when $\bar{A}$ is positive definite [Fong 2011]. Otherwise, we prefer the condition L-curve approach in [Calvetti et al. 2000], which graphs $\text{cond}(T_k)$ against $\|r_k\|$. Yet another L-curve feasible in MINRES-QLP is $\|x_{k-2}^{(2)}\|$ against $\|r_k\|$, since the former is also monotonic (but available two iterations in lag); see Section 2.4.

2. MATHEMATICAL BACKGROUND

Notation and details of algorithmic development from [Choi 2006; Choi et al. 2011] are summarized here.

2.1 Lanczos Process

MINRES and MINRES-QLP use the symmetric Lanczos process [Lanczos 1950] to reduce A to a tridiagonal form $\underline{T_k}$. The process is initialised with $v_0 \equiv 0$, $\beta_1 = \|b\|$, and $\beta_1 v_1 = b$. After k steps of the tridiagonalization, we have produced

$$p_k = Av_k - \sigma v_k, \qquad \alpha_k = v_k^T p_k, \qquad \beta_{k+1} v_{k+1} = p_k - \alpha_k v_k - \beta_k v_{k-1}, \tag{5}$$

where we choose $\beta_k > 0$ to give $\|v_k\| = 1$. Numerically,

$$p_k = Av_k - \sigma v_k - \beta_k v_{k-1}, \qquad \alpha_k = v_k^T p_k, \qquad \beta_{k+1} v_{k+1} = p_k - \alpha_k v_k$$

is slightly better than (5) [Paige 1976], but we can express (5) in matrix form:

$$V_k \equiv \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}, \qquad AV_k = V_{k+1}\underline{T_k}, \qquad \underline{T_k} \equiv \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}, \tag{6}$$

where $T_k = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$, $i = 1, \ldots, k$. In exact arithmetic, the Lanczos vectors in the columns of V_k are orthonormal, and the process stops with $k = \ell$ when $\beta_{\ell+1} = 0$ for some $\ell \leq n$, and then $AV_\ell = V_\ell T_\ell$. The rank of T_ℓ could be ℓ or $\ell - 1$ (see Theorem 2.2).

2.2 MINRES Phase

MINRES-QLP typically starts with a MINRES phase, which applies a series of reflectors Q_k to transform $\underline{T_k}$ to an upper triangular matrix $\underline{R_k}$:

$$Q_k \begin{bmatrix} \underline{T_k} & \beta_1 e_1 \end{bmatrix} = \begin{bmatrix} R_k & t_k \\ 0 & \phi_k \end{bmatrix} \equiv \begin{bmatrix} \underline{R_k} & \bar{t}_{k+1} \end{bmatrix}, \tag{7}$$

where

$$Q_k = Q_{k,k+1} \begin{bmatrix} Q_{k-1} & \\ & 1 \end{bmatrix}, \qquad Q_{k,k+1} \equiv \begin{bmatrix} I_{k-1} & & \\ & c_k & s_k \\ & s_k & -c_k \end{bmatrix}.$$

In the kth step, $Q_{k,k+1}$ is effectively a Householder reflector of dimension 2 [Trefethen and Bau 1997, Exercise 10.4]; and its action including its effect on later columns of T_j, $k < j \leq \ell$, is compactly described by

$$\begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} \left[\begin{array}{ccc|c} \gamma_k & \delta_{k+1} & 0 & \phi_{k-1} \\ \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} & 0 \end{array}\right] = \left[\begin{array}{ccc|c} \gamma_k^{(2)} & \delta_{k+1}^{(2)} & \epsilon_{k+2} & \tau_k \\ 0 & \gamma_{k+1} & \delta_{k+2} & \phi_k \end{array}\right],$$

where the superscripts with numbers in parentheses indicate the number of times the values have been modified. The kth solution approximation to (3) is then defined to be $x_k = V_k y_k$, where y_k solves the subproblem

$$y_k = \arg\min_{y \in \mathbb{R}^k} \|\underline{T_k} y - \beta_1 e_1\| = \arg\min_{y \in \mathbb{R}^k} \|\underline{R_k} y - \bar{t}_{k+1}\|. \tag{8}$$

When $k < \ell$, R_k is nonsingular and the unique solution of the above subproblem satisfies $R_k y_k = t_k$. Instead of solving for y_k, MINRES solves $R_k^T D_k^T = V_k^T$ by forward substitution, obtaining the last column d_k of D_k at iteration k. At the same time, it updates $x_k \in \mathcal{K}_k(A, b)$ (see Table I for definition) via $x_0 \equiv 0$ and

$$x_k = V_k y_k = D_k R_k y_k = D_k t_k = x_{k-1} + \tau_k d_k, \quad \tau_k \equiv e_k^T t_k, \tag{9}$$

where one can show using $V_k = D_k R_k$ that $d_k = (v_k - \delta_k^{(2)} d_{k-1} - \epsilon_k d_{k-2})/\gamma_k^{(2)}$.

2.3 MINRES-QLP Phase

The MINRES phase transfers to the MINRES-QLP phase when an estimate of the condition number of A exceeds an input parameter *trancond*. Thus, *trancond* $> 1/\varepsilon$ leads to MINRES iterates throughout (where $\varepsilon \approx 10^{-16}$ denotes the floating-point precision), whereas *trancond* $= 1$ generates MINRES-QLP iterates from the start.

Suppose for now that there is no MINRES phase. Then MINRES-QLP applies left reflections as in (7) and a further series of right reflections to transform R_k to a lower triangular matrix $L_k = R_k P_k$, where

$$P_k = P_{1,2} \quad P_{1,3}P_{2,3} \quad \cdots \quad P_{k-2,k}P_{k-1,k},$$

$$P_{k-2,k} = \begin{bmatrix} I_{k-3} & & \\ & c_{k2} & s_{k2} \\ & s_{k2} & -c_{k2} \end{bmatrix}, \qquad P_{k-1,k} = \begin{bmatrix} I_{k-2} & & \\ & c_{k3} & s_{k3} \\ & s_{k3} & -c_{k3} \end{bmatrix}.$$

In the kth step, the actions of $P_{k-2,k}$ and $P_{k-1,k}$ are compactly described by

$$\begin{bmatrix} \gamma_{k-2}^{(5)} & & \epsilon_k \\ \vartheta_{k-1} & \gamma_{k-1}^{(4)} & \delta_k^{(2)} \\ & & \gamma_k^{(2)} \end{bmatrix} \begin{bmatrix} c_{k2} & & s_{k2} \\ & 1 & \\ s_{k2} & & -c_{k2} \end{bmatrix} \begin{bmatrix} 1 & & \\ & c_{k3} & s_{k3} \\ & s_{k3} & -c_{k3} \end{bmatrix}$$

$$= \begin{bmatrix} \gamma_{k-2}^{(6)} & & \\ \vartheta_{k-1}^{(2)} & \gamma_{k-1}^{(4)} & \delta_k^{(3)} \\ \eta_k & & \gamma_k^{(3)} \end{bmatrix} \begin{bmatrix} 1 & & \\ & c_{k3} & s_{k3} \\ & s_{k3} & -c_{k3} \end{bmatrix} = \begin{bmatrix} \gamma_{k-2}^{(6)} & & \\ \vartheta_{k-1}^{(2)} & \gamma_{k-1}^{(5)} & \\ \eta_k & \vartheta_k & \gamma_k^{(4)} \end{bmatrix}. \tag{10}$$

The kth approximate solution to (3) is then defined to be $x_k = V_k y_k = V_k P_k u_k = W_k u_k$, where u_k solves the subproblem

$$u_k \equiv \arg\min_u \|u\| \quad \text{s.t.} \quad u \in \arg\min_{u \in \mathbb{R}^k} \left\| \begin{bmatrix} L_k \\ 0 \end{bmatrix} u - \begin{bmatrix} t_k \\ \phi_k \end{bmatrix} \right\|. \tag{11}$$

For $k < \ell$, R_k and L_k are nonsingular because $\underline{T_k}$ has full column rank by Lemma 2.1 below. It is only when $k = \ell$ and $b \notin \text{range}(A)$ that R_k and L_k are singular with rank $\ell - 1$ by Theorem 2.2, in which case one can show that $\eta_k = \gamma_k^{(3)} = \vartheta_k = \gamma_k^{(4)} = 0$ in (10) and $L_\ell = \left[\begin{smallmatrix} L_{\ell-1} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ with $L_{\ell-1}$ nonsingular. In any case, we need to solve only the nonsingular lower triangular systems $L_k u_k = t_k$ or $L_{\ell-1} u_{\ell-1} = t_{\ell-1}$. Then, u_k and $y_k = P_k u_k$ are the min-length solutions of (11) and (8), respectively.

MINRES-QLP updates x_{k-2} to obtain x_k by short-recurrence orthogonal steps:

$$x_{k-2}^{(2)} = x_{k-3}^{(2)} + \mu_{k-2}^{(3)} w_{k-2}^{(4)}, \text{ where } x_{k-3}^{(2)} \equiv W_{k-3}^{(4)} u_{k-3}^{(3)}, \tag{12}$$

$$x_k = x_{k-2}^{(2)} + \mu_{k-1}^{(2)} w_{k-1}^{(3)} + \mu_k w_k^{(2)}. \tag{13}$$

Here w_j refers to the jth column of $W_k = V_k P_k$, and μ_i is the ith element of u_k.

If this phase is preceded by a MINRES phase of k iterations ($0 < k < \ell$), it starts by transferring the last three vectors d_{k-2}, d_{k-1}, d_k to w_{k-2}, w_{k-1}, w_k, and the solution estimate x_k from (9) to $x_{k-2}^{(2)}$ in (12). This needs the last two rows of $L_k u_k = t_k$ (to give μ_{k-1}, μ_k) and the relations $W_k = D_k L_k$ and $x_{k-2}^{(2)} = x_k - \mu_{k-1} w_{k-1} - \mu_k w_k$. The cheaply available right reflections P_k and the bottom right 3×3 submatrix of L_k (i.e., the last term in (10)) need to have been saved in the MINRES phase in order to facilitate the transfer.

2.4 Norm Estimates and Stopping Conditions

Short-term recurrences are used to estimate the following quantities:

$$\begin{aligned}
\|r_k\| &\approx \phi_k = \phi_{k-1}s_k, & \phi_0 &= \|b\| & (\phi_k \searrow)\\
\|Ar_k\| &\approx \psi_k = \phi_k\|[\gamma_{k+1} \;\; \delta_{k+2}]\|, & & & (\psi_\ell = 0)\\
\|x_k^{(2)}\| &\approx \chi_{k-2}^{(2)} = \|[\chi_{k-3}^{(2)} \;\; \mu_{k-2}^{(3)}]\|, & \chi_{-2} = \chi_{-1} &= 0 & (\chi_{k-2}^{(2)} \nearrow)\\
\|x_k\| &\approx \chi_k = \|[\chi_{k-2}^{(2)} \;\; \mu_{k-1}^{(2)} \;\; \mu_k]\|, & \chi_0 &= 0 & (\chi_\ell = \|x^\dagger\|)\\
\|Ax_k\| &\approx \omega_k = \|[\omega_{k-1} \;\; \tau_k]\|, & \omega_0 &= 0 & (\omega_k \nearrow)\\
\|A\| &\approx \mathcal{A}_k = \max\left\{\mathcal{A}_{k-1}, \|\underline{T_k}e_k\|, \overline{\gamma}_k\right\}, & \mathcal{A}_0 &= 0 & (\mathcal{A}_k \nearrow \|A\|)\\
\operatorname{cond}(A) &\approx \kappa_k = \mathcal{A}_k/\underline{\gamma}_k, & \kappa_0 &= 1 & (\kappa_k \nearrow \operatorname{cond}(A))
\end{aligned}$$

where $\overline{\gamma}_k$ and $\underline{\gamma}_k$ are the largest and smallest absolute values of diagonals of L_k, respectively. The up (down) arrows in parentheses indicate that the quantities are monotonic increasing (decreasing) if such properties exist. The last two estimates tend to their targets from below; see [Choi 2006; Choi et al. 2011] for derivation.

MINRES-QLP has 14 possible stopping conditions in five classes that use the above estimates and optional user-input parameters *itnlim*, *rtol*, *Acondlim*, and *maxxnorm*:

(C1) From Lanczos and the QLP factorization:

$$k = \mathit{itnlim}; \qquad \beta_{k+1} < \varepsilon; \qquad \left|\gamma_k^{(4)}\right| < \varepsilon;$$

(C2) Normwise relative backward errors (NRBE) [Paige and Strakoš 2002]:

$$\|r_k\|/\left(\|A\|\|x_k\| + \|b\|\right) \le \max(\mathit{rtol}, \varepsilon); \qquad \|Ar_k\|/\left(\|A\|\|r_k\|\right) \le \max(\mathit{rtol}, \varepsilon);$$

(C3) Regularization attempts:

$$\operatorname{cond}(A) \ge \min(\mathit{Acondlim}, 0.1/\varepsilon); \qquad \|x_k\| \ge \mathit{maxxnorm};$$

(C4) Degenerate cases:

$$\begin{aligned}
&\beta_1 = 0 \quad \Rightarrow \quad b = 0 \quad \Rightarrow \quad x = 0 \text{ is the solution;}\\
&\beta_2 = 0 \quad \Rightarrow \quad v_2 = 0 \quad \Rightarrow \quad Ab = \alpha_1 b,\\
&\qquad \text{i.e., } b \text{ and } \alpha_1 \text{ are an eigenpair of } A, \text{ and } x = b/\alpha_1 \text{ solves } Ax = b;
\end{aligned}$$

(C5) Erroneous inputs:

$$A \text{ not symmetric}; \qquad M \text{ not symmetric}; \qquad M \text{ not positive definite};$$

where M is a preconditioner to be described in the next section. For symmetry of A, it is not practical to check $e_i^T A e_j = e_j^T A e_i$ for all $i, j = 1, \ldots, n$. Instead, we statistically test whether $z = |x^T(Ay) - y^T(Ax)|$ is sufficiently small for two nonzero n-vectors x and y (e.g., each element in the vectors is drawn from the standard normal distribution). For positive definiteness of M, since M is positive definite if and only if M^{-1} is positive definite, we simply test that $z_k^T M^{-1} z_k = z_k^T q_k > 0$ each iteration (see Section 3).

We find that the recurrence relations for ϕ_k and ψ_k hold to high accuracy. Thus x_k is an acceptable solution of (3) if the computed value of ϕ_k or ψ_k is suitably small according to the NRBE tests in class (C2) above. When a condition in (C3) is met, the final x_k may or may not be an acceptable solution.

The class (C1) tests for small β_{k+1} and $\gamma_k^{(4)}$ are included in the unlikely case in practice that the theoretical Lanczos termination occurs. Ideally one of the NRBE tests should cause MINRES-QLP to terminate. If not, it is an indication that the problem is very ill-conditioned, in which case the regularization and preconditioning techniques of Sections 1.2 and 3 may be helpful.

2.5 Two Theorems

We complete this section by presenting two theorems from [Choi et al. 2011] with slightly simpler proofs.

LEMMA 2.1. $\operatorname{rank}(\underline{T_k}) = k$ *for all* $k < \ell$.

PROOF. For $k < \ell$ we have $\beta_1, \ldots, \beta_{k+1} > 0$ by definition. Hence $\underline{T_k}$ has full column rank. □

THEOREM 2.2. T_ℓ *is nonsingular if and only if* $b \in \operatorname{range}(A)$. *Furthermore,* $\operatorname{rank}(T_\ell) = \ell - 1$ *if* $b \notin \operatorname{range}(A)$.

PROOF. We use $AV_\ell = V_\ell T_\ell$ twice. First, if T_ℓ is nonsingular, we can solve $T_\ell y_\ell = \beta_1 e_1$ and then $AV_\ell y_\ell = V_\ell T_\ell y_\ell = V_\ell \beta_1 e_1 = b$. Conversely, if $b \in \operatorname{range}(A)$, then $\operatorname{range}(V_\ell) \subseteq \operatorname{range}(A)$. Suppose T_ℓ is singular. Then there exists $z \neq 0$ such that $V_\ell T_\ell z = AV_\ell z = 0$. That is, $0 \neq V_\ell z \in \operatorname{null}(A)$. But this is impossible because $V_\ell z \in \operatorname{range}(A)$ and $\operatorname{null}(A) \cap \operatorname{range}(V_\ell) = 0$. Thus, T_ℓ must be nonsingular.

We have shown that if $b \notin \operatorname{range}(A)$, $T_\ell = \begin{bmatrix} \underline{T_{\ell-1}} & \beta_\ell e_{\ell-1} \\ & \alpha_\ell \end{bmatrix}$ is singular, and therefore $\ell > \operatorname{rank}(T_\ell) \geq \operatorname{rank}(\underline{T_{\ell-1}}) = \ell - 1$ by Lemma 2.1. Therefore, $\operatorname{rank}(T_\ell) = \ell - 1$. □

By Lemma 2.1 and Theorem 2.2 we are assured that the QLP decomposition without column pivoting [Stewart 1999; Choi et al. 2011] for $\underline{T_k}$ is rank-revealing, which is a necessary precondition for solving a least-squares problem.

THEOREM 2.3. *In MINRES-QLP,* x_ℓ *is the minimum-length solution of (3).*

PROOF. y_ℓ comes from the min-length LS solution of $T_\ell y_\ell \approx \beta_1 e_1$ and thus satisfies the normal equation $T_\ell^2 y_\ell = T_\ell \beta_1 e_1$ and $y_\ell \in \operatorname{range}(T_\ell)$. Now $x_\ell = V_\ell y_\ell$ and $Ax_\ell = AV_\ell y_\ell = V_\ell T_\ell y_\ell$. Hence $A^2 x_\ell = AV_\ell T_\ell y_\ell = V_\ell T_\ell^2 y_\ell = V_\ell T_\ell \beta_1 e_1 = Ab$. Thus x_ℓ is an LS solution of (3). Since $y_\ell \in \operatorname{range}(T_\ell)$, $y_\ell = T_\ell z$ for some z, and so $x_\ell = V_\ell y_\ell = V_\ell T_\ell z = AV_\ell z \in \operatorname{range}(A)$ is the min-length LS solution of (3). □

3. PRECONDITIONING

Iterative methods can be accelerated if preconditioners are available and well-chosen. For MINRES-QLP, we want to choose a symmetric positive-definite matrix M to solve a nonsingular system (1) by implicitly solving an equivalent symmetric consistent system $M^{-\frac{1}{2}} A M^{-\frac{1}{2}} \bar{x} = \bar{b}$, where $M^{\frac{1}{2}} x = \bar{x}$, $\bar{b} = M^{-\frac{1}{2}} b$, and $\operatorname{cond}(M^{-\frac{1}{2}} A M^{-\frac{1}{2}}) \ll \operatorname{cond}(A)$. This two-sided preconditioning preserves symmetry. Thus we can derive preconditioned MINRES-QLP by applying MINRES-QLP to the equivalent problem and obtain $x = M^{-\frac{1}{2}} \bar{x}$.

With preconditioned MINRES-QLP, we can solve a singular consistent system (2), but we will obtain a least-squares solution that is not necessarily the minimum-length solution (unless $M = I$). For inconsistent systems (3), preconditioning alters the least-squares norm to $\|\cdot\|_{M^{-1}}$, and the solution is of minimum length in the new norm space. We refer readers to [Choi et al. 2011, Section 7] for a detailed discussion of various approaches to preserving the two-norm "minimum length."

To derive MINRES-QLP, we define

$$z_k = \beta_k M^{\frac{1}{2}} v_k, \qquad q_k = \beta_k M^{-\frac{1}{2}} v_k, \qquad \text{so that} \quad Mq_k = z_k. \tag{14}$$

Then $\beta_k = \|\beta_k v_k\| = \|M^{-\frac{1}{2}} z_k\| = \|z_k\|_{M^{-1}} = \|q_k\|_M = \sqrt{q_k^T z_k}$, where the square root is well defined because M is positive definite, and the following expressions replace the quantities in (5) in the Lanczos iterations:

$$p_k = Aq_k - \sigma q_k, \qquad \alpha_k = \frac{1}{\beta_k^2} q_k^T p_k, \qquad z_{k+1} = \frac{1}{\beta_k} p_k - \frac{\alpha_k}{\beta_k} z_k - \frac{\beta_k}{\beta_{k-1}} z_{k-1}. \tag{15}$$

We also need to solve the system $Mq_k = z_k$ in (14) at each iteration.

In the MINRES phase, we define $\bar{d}_k = M^{-\frac{1}{2}} d_k$ and update the solution of the original problem (1) by

$$\bar{d}_k = \Big(\frac{1}{\beta_k} q_k - \delta_k^{(2)} \bar{d}_{k-1} - \epsilon_k \bar{d}_{k-2}\Big) \big/ \gamma_k^{(2)}, \qquad x_k = M^{-\frac{1}{2}} \bar{x}_k = x_{k-1} + \tau_k \bar{d}_k.$$

In the MINRES-QLP phase, we define $\overline{W}_k \equiv M^{-\frac{1}{2}} W_k = (M^{-\frac{1}{2}} V_k) P_k$ and update the solution estimate of problem (1) by orthogonal steps:

$$\begin{aligned}
\bar{w}_k &= -(c_{k2}/\beta_k) q_k + s_{k2} \bar{w}_{k-2}^{(3)}, & \bar{w}_{k-2}^{(4)} &= (s_{k2}/\beta_k) q_k + c_{k2} \bar{w}_{k-2}^{(3)}, \\
\bar{w}_k^{(2)} &= s_{k3} \bar{w}_{k-1}^{(2)} - c_{k3} \bar{w}_k, & \bar{w}_{k-1}^{(3)} &= c_{k3} \bar{w}_{k-1}^{(2)} + s_{k3} \bar{w}_k, \\
x_{k-2}^{(2)} &= x_{k-3}^{(2)} + \mu_{k-2}^{(3)} \bar{w}_{k-2}^{(4)}, & x_k &= x_{k-2}^{(2)} + \mu_{k-1}^{(2)} \bar{w}_{k-1}^{(3)} + \mu_k \bar{w}_k^{(2)}.
\end{aligned}$$

Let $\bar{r}_k = \bar{b} - M^{-\frac{1}{2}} A M^{-\frac{1}{2}} \bar{x}_k = M^{-\frac{1}{2}} r_k$. Then $x_k = M^{-\frac{1}{2}} \bar{x}_k$ is an acceptable solution of (1) if the computed value of $\phi_k \approx \|\bar{r}_k\| = \|r_k\|_{M^{-1}}$ is sufficiently small.

We can now present our pseudocode in Algorithm 1. The reflectors are implemented in Algorithm 2 SymOrtho(a, b) for real a and b, which is a stable form for computing $r = \sqrt{a^2 + b^2} \geq 0$, $c = \frac{a}{r}$, and $s = \frac{b}{r}$. The complexity is at most 6 flops and a square root. Algorithm 1 lists all steps of MINRES-QLP with preconditioning. For simplicity, $\bar{w}_k$ is written as w_k for all relevant k. Also, the output x solves $Ax \approx b$, but other outputs are associated with the preconditioned system.

4. KEY FORTRAN 90 DESIGN FEATURES

Our FORTRAN 90 package contains the following files for symmetric problems with the first three files forming the core. Their dependencies are depicted in Figure 1.

1. `minresqlpDataModule.f90`: defines precision and constants used in other modules
2. `minresqlpBlasModule.f90`: packages some BLAS functions [Burkardt]
3. `minresqlpModule.f90`: implements MINRES-QLP with preconditioning

Algorithm 1: Pseudocode of preconditioned MINRES-QLP for solving $(A-\sigma I)x \approx b$. In the right-justified comments, $\tilde{A} \equiv M^{-\frac{1}{2}}(A-\sigma I)M^{-\frac{1}{2}}$.

input: A, b, σ, M

$z_0 = 0, \quad z_1 = b, \quad$ Solve $Mq_1 = z_1, \quad \beta_1 = \sqrt{b^T q_1}, \quad \phi_0 = \beta_1$ `[Initialize]`

$w_0 = w_{-1} = 0, \quad x_{-2} = x_{-1} = x_0 = 0$

$c_{0,1} = c_{0,2} = c_{0,3} = -1, \quad s_{0,1} = s_{0,2} = s_{0,3} = 0, \quad \tau_0 = \omega_0 = \chi_{-2} = \chi_{-1} = \chi_0 = 0$

$\kappa_0 = 1, \quad \mathcal{A}_0 = \delta_1 = \gamma_{-1} = \gamma_0 = \eta_{-1} = \eta_0 = \eta_1 = \vartheta_{-1} = \vartheta_0 = \vartheta_1 = \mu_{-1} = \mu_0 = 0$

$k = 0$

while no stopping condition is satisfied **do**

- $k \leftarrow k + 1$
- $p_k = Aq_k - \sigma q_k, \quad \alpha_k = \frac{1}{\beta_k^2} q_k^T p_k$ `[Preconditioned Lanczos]`
- $z_{k+1} = \frac{1}{\beta_k} p_k - \frac{\alpha_k}{\beta_k} z_k - \frac{\beta_k}{\beta_{k-1}} z_{k-1}$
- Solve $Mq_{k+1} = z_{k+1}, \quad \beta_{k+1} = \sqrt{q_{k+1}^T z_{k+1}}$
- **if** $k = 1$ **then** $\rho_k = \|[\alpha_k \;\; \beta_{k+1}]\|$ **else** $\rho_k = \|[\beta_k \;\; \alpha_k \;\; \beta_{k+1}]\|$
- $\delta_k^{(2)} = c_{k-1,1}\delta_k + s_{k-1,1}\alpha_k$ `[Previous left reflection...]`
- $\gamma_k = s_{k-1,1}\delta_k - c_{k-1,1}\alpha_k$ `[on middle two entries of` $\underline{T_k} e_k$ `...]`
- $\epsilon_{k+1} = s_{k-1,1}\beta_{k+1}$ `[produces first two entries in` $\underline{T_{k+1}} e_{k+1}$`]`
- $\delta_{k+1} = -c_{k-1,1}\beta_{k+1}$
- $c_{k1}, s_{k1}, \gamma_k^{(2)} \leftarrow \mathrm{SymOrtho}(\gamma_k, \beta_{k+1})$ `[Current left reflection]`
- $c_{k2}, s_{k2}, \gamma_{k-2}^{(6)} \leftarrow \mathrm{SymOrtho}(\gamma_{k-2}^{(5)}, \epsilon_k)$ `[First right reflection]`
- $\delta_k^{(3)} = s_{k2}\vartheta_{k-1} - c_{k2}\delta_k^{(2)}, \quad \gamma_k^{(3)} = -c_{k2}\gamma_k^{(2)}, \quad \eta_k = s_{k2}\gamma_k^{(2)}$
- $\vartheta_{k-1}^{(2)} = c_{k2}\vartheta_{k-1} + s_{k2}\delta_k^{(2)}$
- $c_{k3}, s_{k3}, \gamma_{k-1}^{(5)} \leftarrow \mathrm{SymOrtho}(\gamma_{k-1}^{(4)}, \delta_k^{(3)})$ `[Second right reflection...]`
- $\vartheta_k = s_{k3}\gamma_k^{(3)}, \quad \gamma_k^{(4)} = -c_{k3}\gamma_k^{(3)}$ `[to zero out` $\delta_k^{(3)}$`]`
- $\tau_k = c_{k1}\phi_{k-1}$ `[Last element of` t_k`]`
- $\phi_k = s_{k1}\phi_{k-1}, \quad \psi_{k-1} = \phi_{k-1}\|[\gamma_k \;\; \delta_{k+1}]\|$ `[Update` $\|\bar{r}_k\|$`,` $\|\tilde{A}\bar{r}_{k-1}\|$`]`
- **if** $k = 1$ **then** $\gamma_{\min} = \gamma_1$ **else** $\gamma_{\min} \leftarrow \min\{\gamma_{\min}, \gamma_{k-2}^{(6)}, \gamma_{k-1}^{(5)}, |\gamma_k^{(4)}|\}$
- $\mathcal{A}_k = \max\{\mathcal{A}_{k-1}, \rho_k, \gamma_{k-2}^{(6)}, \gamma_{k-1}^{(5)}, |\gamma_k^{(4)}|\}$ `[Update` $\|\tilde{A}\|$`]`
- $\omega_k = \|[\omega_{k-1} \;\; \tau_k]\|, \quad \kappa_k \leftarrow \mathcal{A}_k/\gamma_{\min}$ `[Update` $\|\tilde{A}x_k\|$`,` $\mathrm{cond}(\tilde{A})$`]`
- $w_k = -(c_{k2}/\beta_k)q_k + s_{k2}w_{k-2}^{(3)}$ `[Update` w_{k-2}`,` w_{k-1}`,` w_k`]`
- $w_{k-2}^{(4)} = (s_{k2}/\beta_k)q_k + c_{k2}w_{k-2}^{(3)}$
- **if** $k > 2$ **then** $w_k^{(2)} = s_{k3}w_{k-1}^{(2)} - c_{k3}w_k, \quad w_{k-1}^{(3)} = c_{k3}w_{k-1}^{(2)} + s_{k3}w_k$
- **if** $k > 2$ **then** $\mu_{k-2}^{(3)} = (\tau_{k-2} - \eta_{k-2}\mu_{k-4}^{(4)} - \vartheta_{k-2}\mu_{k-3}^{(3)})/\gamma_{k-2}^{(6)}$ `[Update` μ_{k-2}`]`
- **if** $k > 1$ **then** $\mu_{k-1}^{(2)} = (\tau_{k-1} - \eta_{k-1}\mu_{k-3}^{(3)} - \vartheta_{k-1}^{(2)}\mu_{k-2}^{(3)})/\gamma_{k-1}^{(5)}$ `[Update` μ_{k-1}`]`
- **if** $\gamma_k^{(4)} \neq 0$ **then** $\mu_k = (\tau_k - \eta_k\mu_{k-2}^{(3)} - \vartheta_k\mu_{k-1}^{(2)})/\gamma_k^{(4)}$ **else** $\mu_k = 0$ `[Compute` μ_k`]`
- $x_{k-2}^{(2)} = x_{k-3}^{(2)} + \mu_{k-2}^{(3)}w_{k-2}^{(3)}$ `[Update` x_{k-2}`]`
- $x_k = x_{k-2}^{(2)} + \mu_{k-1}^{(2)}w_{k-1}^{(3)} + \mu_k w_k^{(2)}$ `[Compute` x_k`]`
- $\chi_{k-2}^{(2)} = \|[\chi_{k-3}^{(2)} \;\; \mu_{k-2}^{(3)}]\|$ `[Update` $\|x_{k-2}\|$`]`
- $\chi_k = \|[\chi_{k-2}^{(2)} \;\; \mu_{k-1}^{(2)} \;\; \mu_k]\|$ `[Compute` $\|x_k\|$`]`

$x = x_k, \quad \phi = \phi_k, \quad \psi = \phi_k\|[\gamma_{k+1} \;\; \delta_{k+2}]\|, \quad \chi = \chi_k, \quad \mathcal{A} = \mathcal{A}_k, \quad \omega = \omega_k$

output: $x, \phi, \psi, \chi, \mathcal{A}, \kappa, \omega$

Algorithm 2: Algorithm SymOrtho.

input: a, b

if $b = 0$ **then** $s = 0, \quad r = |a|$
  if $a = 0$ **then** $c = 1$ **else** $c = \text{sign}(a)$
else if $a = 0$ **then**
  $c = 0, \quad s = \text{sign}(b), \quad r = |b|$
else if $|b| \geq |a|$ **then**
  $\tau = a/b, \quad s = \text{sign}(b)/\sqrt{1+\tau^2}, \quad c = s\tau, \quad r = b/s$
else if $|a| > |b|$ **then**
  $\tau = b/a, \quad c = \text{sign}(a)/\sqrt{1+\tau^2}, \quad s = c\tau, \quad r = a/c$

output: c, s, r

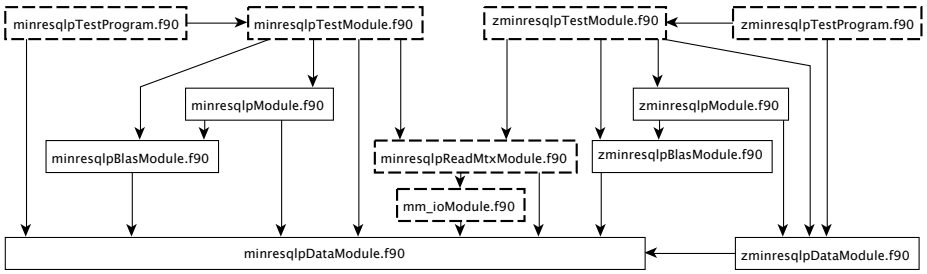

Fig. 1. FORTRAN 90 source files and their dependencies. Filenames boxed in broken lines are optional, and the corresponding files are used mainly for testing and demonstration.

4. `mm_ioModule.f90` and `minresqlpReadMtxModule.f90`: packages subroutines for reading Matrix Market files [Matrix Market; Burkardt]
5. `minresqlpTestModule.f90`: illustrates how MINRES-QLP can call `Aprod` or `Msolve` with a short fixed parameter list, even if it needs arbitrary other data
6. `minresqlpTestProgram.f90`: contains the main driver program for unit tests
7. `Makefile`: compiles the FORTRAN source files via the Unix command `make`
8. `minresqlp_f90.README`: contains information about software license, other files in the package, and program compilation and execution.

The counterparts of these programs for Hermitian problems have the same filenames prefixed with the letter "`z`".

In our FORTRAN 90 implementation, we use ***`modules`*** instead of the obsolete FORTRAN 77 `COMMON` blocks for grouping programs units and data together and controlling their availability to other program units. A module can use `public` data and subroutines from other modules (by declaring an ***`interface`*** block), share its own `public` data and subroutines with other program units, and hide its own `private` data and subroutines from being used by other program units.

In `minresqlpModule.f90` we define a `public` subroutine `MINRESQLP` that implements Algorithm 1. Two input arguments of this subroutine, `Aprod` and `Msolve`, are external user-defined subroutines—we recommend they be `private` for data integrity. The subroutine `Aprod` defines the matrix A as an operator. For a given

vector x, the FORTRAN statement `call Aprod(n, x, y)` must return the product $y = Ax$ without altering x. The subroutine `Msolve` is optional, and it defines a symmetric positive-definite matrix as an operator M that serves as a preconditioner. For a given vector y, the FORTRAN statement `call MSolve(n, y, x)` must solve the linear system $Mx = y$ without altering y. To provide the compiler the necessary information about these `private` subroutines defined in `minresqlpTestModule`, an `interface` block in subroutine `MINRESQLP` is declared, which essentially replicates the headers of `Aprod` and `Msolve` in `minresqlpTestModule`.

A public routine `minresqlptest`, also defined in module `minresqlpTestModule`, calls `MINRESQLP` with `Aprod` and `Msolve` passed to `MINRESQLP` as parameters.

We declare all data variables in `minresqlpTestModule` used for defining `Aprod` and `Msolve` to be `private` so that they are accessible to all the subroutines in the module but not outside.

To summarize, we have described and provided a pattern that allows MINRES-QLP users to solve different problems by simply editing `minresTestModule` (and possibly the main program `minresTestProgram`, which calls `minresqlptest`). Users do not need to change `MINRESQLP` as long as the header of subroutines `Aprod` and `Msolve` stay the same in `minresTestModule`.

Our design spares users from implementing *reverse communication*, and hence enables the development of iterative methods without *a priori* knowledge of users' problem data A and M (by returning control to the calling program every time `Aprod` or `Msolve` is to be invoked). While reverse communication is widely used in scientific computing with FORTRAN 77, the resulting code usually appears formidable and unrecognizable from the original pseudocode; see [Dongarra et al. 1995] and [Oliveira and Stewart 2006] for two examples of CG and numerical integration coded in FORTRAN 77 and 90, respectively. Our MINRES-QLP implementation achieves the purpose of reverse communication while preserving code readability and thus maintainability. The FORTRAN 90 module structure allows a user's Ax products and $Mx = y$ solves to be implemented outside MINRES-QLP in the same way that MATLAB's function handles operate.

Finally, unit testing is an important software development strategy that cannot be overemphasized, especially in the scientific computing communities. Unit testing usually consists of multiple small and fast but specific and illuminating test cases that check whether the code behaves as designed. Software development is incremental, and errors (also known as bugs) are often found over time. Adding new functionalities or fixing errors often breaks the code for some earlier successful test cases. It is therefore critical to expand the test cases and to ensure that all unit tests are executed with expected results every time a program unit is updated.

In our development of FORTRAN 90 MINRES-QLP, we have created a suite of 52 test cases including singular matrices representative of real-world applications [Foster 2009; Davis and Hu 2011]. The test program outputs results to `MINRESQLP.txt`. If users need to modify subroutine `MINRESQLP`, they can run these test cases and search for the word "`appear`" in the output file to check whether all tests are reported to be successful. For more sophisticated unit testing frameworks employed in large-scale scientific software development, see [O'Boyle et al. 2008].

Further details on interface and implementation, with additional numerical examples and documentation, are given in [Choi and Saunders 2012].

As a last note, careful choices of parameter values are critical in the convergence behavior of iterative solvers. While the default parameter values in MINRES-QLP work well in most tests, they may need to be fine-tuned in some cases by trial and error, solving a series of problems as in iterative regularization, or partial or full reorthogonalization of the Lanczos vectors.

ACKNOWLEDGMENTS

We thank Christopher Paige for his contribution to the theory of MINRES-QLP [Choi et al. 2011]. We also thank Tim Hopkins and David Saunders for testing and running our FORTRAN 90 package on the Intel ifort compiler and the NAG Fortran compiler, resulting in more robust code. We are grateful to Zhaojun Bai and both anonymous reviewers for their patience and constructive comments. The first author also thanks Jed Brown, Ian Foster, Todd Munson, Gail Pieper, and Stefan Wild for their feedback and support during the development of this work. We express our gratitude to the SIAM 2012 SIAG/LA Prize Committee for their favorable consideration of MINRES-QLP [Choi et al. 2011].

REFERENCES

Burkardt, J. FORTRAN90 software. `http://people.sc.fsu.edu/~jburkardt/`.

Calvetti, D., Lewis, B., and Reichel, L. 2000. An L-curve for the MINRES method. In *Proceedings of SPIE*. Vol. 4116. 385–395.

Choi, S.-C. T. 2006. Iterative Methods for Singular Linear Equations and Least-Squares Problems. Ph.D. thesis, ICME, Stanford University, CA.

Choi, S.-C. T., Paige, C. C., and Saunders, M. A. 2011. MINRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems. *SIAM J. Sci. Comput. 33,* 4, 1810–1836.

Choi, S.-C. T. and Saunders, M. A. 2012. ALGORITHM & DOCUMENTATION: MINRES-QLP for singular symmetric and Hermitian linear equations and least-squares problems. Technical Report ANL/MCS-P3027-0812, Computation Institute, University of Chicago, IL.

Davis, T. A. and Hu, Y. 2011. The University of Florida sparse matrix collection. *ACM Trans. Math. Software 38,* 1, 1:1–1:25.

Dongarra, J., Eijkhout, V., and Kalhan, A. 1995. Reverse communication interface for linear algebra templates for iterative methods. Report UT-CS-95-291, University of Tennessee, TN.

Fong, D. C.-L. 2011. Minimum-Residual Methods for Sparse Least-Squares Using Golub-Kahan Bidiagonalization. Ph.D. thesis, ICME, Stanford University, CA.

Fong, D. C.-L. and Saunders, M. A. 2011. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput. 33*, 2950–2971.

Fong, D. C.-L. and Saunders, M. A. 2012. CG versus MINRES: An empirical comparison. *SQU Journal for Science 17,* 1, 44–62.

Foster, L. 2009. San Jose State University Singular Matrix Database. `http://www.math.sjsu.edu/singular/matrices/`.

Freund, R. W. and Nachtigal, N. M. 1994. A new Krylov-subspace method for symmetric indefinite linear systems. In *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, W. F. Ames, Ed. IMACS, 1253–1256.

Golub, G. H. and Van Loan, C. F. 1996. *Matrix Computations*, 3rd ed. Johns Hopkins University Press, Baltimore, MD.

Hansen, P. C. 1998. *Rank-deficient and Discrete Ill-posed Problems.* SIAM Monographs on Mathematical Modeling and Computation. SIAM, Philadelphia, PA.

Hestenes, M. R. and Stiefel, E. 1952. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards 49*, 409–436.

Lanczos, C. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards 45*, 255–282.

Matrix Market. Text File Formats. `http://math.nist.gov/MatrixMarket/formats.html`.

O'Boyle, N. M., Tenderholt, A. L., and Langner, K. M. 2008. cclib: A library for package-independent computational chemistry algorithms. *J. Comput. Chem. 29,* 5, 839–845.

Oliveira, S. and Stewart, D. 2006. *Writing Scientific Software: A Guide to Good Style*. Cambridge University Press, Cambridge, UK.

Paige, C. C. 1976. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Math. Appl. 18,* 3, 341–349.

Paige, C. C. and Saunders, M. A. 1975. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal. 12,* 4, 617–629.

Paige, C. C. and Saunders, M. A. 1982a. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software 8,* 1, 43–71.

Paige, C. C. and Saunders, M. A. 1982b. Algorithm 583; LSQR: Sparse linear equations and least-squares problems. *ACM Trans. Math. Software 8,* 2, 195–209.

Paige, C. C. and Strakoš, Z. 2002. Residual and backward error bounds in minimum residual Krylov subspace methods. *SIAM J. Sci. Comput. 23,* 6, 1899–1924.

Saad, Y. and Schultz, M. H. 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput. 7,* 3, 856–869.

SOL. Systems Optimization Laboratory, Stanford University. `www.stanford.edu/group/SOL`.

Stewart, G. W. 1999. The QLP approximation to the singular value decomposition. *SIAM J. Sci. Comput. 20,* 4, 1336–1348.

Trefethen, L. N. and Bau, III, D. 1997. *Numerical Linear Algebra*. SIAM, Philadelphia, PA.

Received 03 Aug 2011; accepted 15 Jan 2012; revised January 15, 2013.